7. 웹 스크래핑

웹 서버에 요청하고 응답받기


```
import requests

url = "https://www.python.org/"
resp = requests.get(url)
print(resp)

url2 = "https://www.python.org/1"
resp2 = requests.get(url2)
print(resp2)
```

```
<Response [200]>
<Response [404]>
```

웹 페이지 소스코드 확인하기


```
import requests

url = "https://www.python.org/"
resp = requests.get(url)

html = resp.text
print(html)
```

로봇 배제 표준(robots.txt)

```
import requests

urls = ["https://www.naver.com/", "https://www.python.org/"]
filename= "robots.txt"

for url in urls:
    file_path = url + filename
    print(file_path)
    resp = requests.get(file_path)
    print(resp.text)
    print("\n")
```

```
https://www.naver.com/robots.txt
User-agent: *
Disallow: /
Allow : /$

https://www.python.org/robots.txt
# Directions for robots.  See this URL:
# http://www.robotstxt.org/robotstxt.html
# for a description of the file format.
```

로봇 배제 표준(robots.txt)

```
User-agent: HTTrack
User-agent: puf
User-agent: MSIECrawler
Disallow: /

# The Krugle web crawler (though based on Nutch) is OK.
User-agent: Krugle
Allow: /
Disallow: /~guido/orlijn/
Disallow: /webstats/

# No one should be crawling us with Nutch.
User-agent: Nutch
Disallow: /

# Hide old versions of the documentation and various large sets of files.
User-agent: *
Disallow: /~guido/orlijn/
Disallow: /webstats/
```

BeautifulSoup 객체 만들기

```
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
resp = requests.get(url)
html_src = resp.text

soup = BeautifulSoup(html_src, 'html.parser')
print(type(soup))
print("\n")

print(soup.head)
print("\n")
print(soup.body)
print("\n")

print('title 태그 요소: ', soup.title)
print('title 태그 이름: ', soup.title.name)
print('title 태그 문자열: ', soup.title.string)
```

BeautifulSoup 객체 만들기

```
<class 'bs4.BeautifulSoup'>

<head>
<meta charset="utf-8"/>
...
</script></body>

title 태그 요소:  <title>Seoul Metropolitan Subway - Wikipedia</title>
title 태그 이름:  title
title 태그 문자열:  Seoul Metropolitan Subway - Wikipedia
```

```
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
resp = requests.get(url)
html_src = resp.text

soup = BeautifulSoup(html_src, 'html.parser')

first_img = soup.find(name='img')
print(first_img)
print("\n")

target_img = soup.find(name='img', attrs={'alt':'Seoul-Metro-2004-
20070722.jpg'})
print(target_img)
```

```
<img alt="South Korea subway logo.svg" data-file-height="450" data-file-
width="450" decoding="async" height="75"
src="//upload.wikimedia.org/wikipedia/commons/thumb/1/12/South_Korea_subwa
y_logo.svg/75px-South_Korea_subway_logo.svg.png"
srcset="//upload.wikimedia.org/wikipedia/commons/thumb/1/12/South_Korea_su
bway_logo.svg/113px-South_Korea_subway_logo.svg.png 1.5x,
//upload.wikimedia.org/wikipedia/commons/thumb/1/12/South_Korea_subway_log
o.svg/150px-South_Korea_subway_logo.svg.png 2x" width="75"/>


<img alt="Seoul-Metro-2004-20070722.jpg" data-file-height="2100" data-
file-width="2800" decoding="async" height="169"
src="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-2004-
20070722.jpg/225px-Seoul-Metro-2004-20070722.jpg"
srcset="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-
2004-20070722.jpg/338px-Seoul-Metro-2004-20070722.jpg 1.5x,
//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-2004-
20070722.jpg/450px-Seoul-Metro-2004-20070722.jpg 2x" width="225"/>
```

웹 문서의 그림 이미지 파일을 PC에 저장하기


```
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
resp = requests.get(url)
html_src = resp.text

soup = BeautifulSoup(html_src, 'html.parser')

target_img = soup.find(name='img', attrs={'alt':'Seoul-Metro-2004-20070722.jpg'})
print('HTML 요소: ', target_img)
print("\n")

target_img_src = target_img.get('src')
print('이미지 파일 경로: ', target_img_src)
print("\n")

target_img_resp = requests.get('http:' + target_img_src)
out_file_path = "download_image.jpg"

with open(out_file_path, 'wb') as out_file:
    out_file.write(target_img_resp.content)
    print("이미지 파일로 저장하였습니다.")
```

웹 문서의 그림 이미지 파일을 PC에 저장하기

HTML **요소**:

이미지 파일 경로: //upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-2004-20070722.jpg/225px-Seoul-Metro-2004-20070722.jpg

이미지 파일로 저장하였습니다.

웹 문서에 포함된 모든 하이퍼링크 추출하기


```
import requests, re
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
resp = requests.get(url)
html_src = resp.text
soup = BeautifulSoup(html_src, 'html.parser')

links = soup.find_all("a")
print("하이퍼링크의 개수: ", len(links))
print("\n")
print("첫 3개의 원소: ", links[:3])
print("\n")

wiki_links = soup.find_all(name="a", href=re.compile("/wiki/"), limit=3)
print("/wiki/ 문자열이 포함된 하이퍼링크: ", wiki_links)
print("\n")

external_links = soup.find_all(name="a", attrs={"class":"external text"}, limit=3)
print("class 속성으로 추출한 하이퍼링크: ", external_links)
```

웹 문서에 포함된 모든 하이퍼링크 추출하기

하이퍼링크의 개수: 968

첫 3개의 원소: [, Jump to navigation, Jump to search]

/wiki/ **문자열이 포함된 하이퍼링크**: [, , Line 2]

웹 문서에 포함된 모든 하이퍼링크 추출하기


```
class 속성으로 추출한 하이퍼링크:  [<a class="external text"
href="http://www.seoulmetro.co.kr/kr/board.do?menuIdx=548" rel="nofollow">"자료실 : 알림마
당&gt;자료실&gt;자료실"</a>, <a class="external text"
href="http://www.korail.com/file/statistics/2012/2012-04.pdf" rel="nofollow">2012 Korail
Statistics</a>, <a class="external text"
href="https://web.archive.org/web/20140227072212/http://www.korail.com/file/statistics/20
12/2012-04.pdf" rel="nofollow">Archived</a>]
```

7. 웹 스크래핑

7.1 웹 스크래핑 I

7.2 웹 스크래핑 II

7.3 웹 스크래핑 III

```
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
resp = requests.get(url)
html_src = resp.text
soup = BeautifulSoup(html_src, 'html.parser')

subway_image = soup.select('#mw-content-text > div > table:nth-child(3) >
\
                           tbody > tr:nth-child(2) > td > a > img')
print(subway_image)
print("\n")
print(subway_image[0])
print("\n")

subway_image2 = soup.select('tr > td > a > img')
print(subway_image2[1])
```

```
[<img alt="Seoul-Metro-2004-20070722.jpg" data-file-height="2100" data-file-width="2800"
decoding="async" height="169" src="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-
2004-20070722.jpg/225px-Seoul-Metro-2004-20070722.jpg"
srcset="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-2004-20070722.jpg/338px-
Seoul-Metro-2004-20070722.jpg 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-
Metro-2004-20070722.jpg/450px-Seoul-Metro-2004-20070722.jpg 2x" width="225"/>]


<img alt="Seoul-Metro-2004-20070722.jpg" data-file-height="2100" data-file-width="2800"
decoding="async" height="169" src="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-
2004-20070722.jpg/225px-Seoul-Metro-2004-20070722.jpg"
srcset="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-2004-20070722.jpg/338px-
Seoul-Metro-2004-20070722.jpg 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-
Metro-2004-20070722.jpg/450px-Seoul-Metro-2004-20070722.jpg 2x" width="225"/>


<img alt="Seoul-Metro-2004-20070722.jpg" data-file-height="2100" data-file-width="2800"
decoding="async" height="169" src="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-
2004-20070722.jpg/225px-Seoul-Metro-2004-20070722.jpg"
srcset="//upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-Metro-2004-20070722.jpg/338px-
Seoul-Metro-2004-20070722.jpg 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/2/29/Seoul-
Metro-2004-20070722.jpg/450px-Seoul-Metro-2004-20070722.jpg 2x" width="225"/>
```

```
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
resp = requests.get(url)
html_src = resp.text
soup = BeautifulSoup(html_src, 'html.parser')

links = soup.select('a')
print(len(links))
print("\n")

print(links[:3])
print("\n")

external_links = soup.select('a[class="external text"]')
print(external_links[:3])
print("\n")
```

CSS Selector 활용하기 2

```
id_selector = soup.select('#siteNotice')
print(id_selector)
print("\n")

id_selector2 = soup.select('div#siteNotice')
print(id_selector2)
print("\n")

id_selector3 = soup.select('p#siteNotice')
print(id_selector3)
print("\n")

class_selector = soup.select('.mw-headline')
print(class_selector)
print("\n")

class_selector2 = soup.select('span.mw-headline')
print(class_selector2)
```

```
968


[<a id="top"></a>, <a class="mw-jump-link" href="#mw-head">Jump to navigation</a>, <a class="mw-
jump-link" href="#p-search">Jump to search</a>]


[<a class="external text" href="http://www.seoulmetro.co.kr/kr/board.do?menuIdx=548"
rel="nofollow">"자료실 : 알림마당&gt;자료실&gt;자료실"</a>, <a class="external text"
href="http://www.korail.com/file/statistics/2012/2012-04.pdf" rel="nofollow">2012 Korail
Statistics</a>, <a class="external text"
href="https://web.archive.org/web/20140227072212/http://www.korail.com/file/statistics/2012/2012-
04.pdf" rel="nofollow">Archived</a>]


[<div class="mw-body-content" id="siteNotice"><!-- CentralNotice --></div>]


[<div class="mw-body-content" id="siteNotice"><!-- CentralNotice --></div>]


[]
```

```
[<span class="mw-headline" id="Overview">Overview</span>, <span class="mw-headline"
id="History">History</span>, <span class="mw-headline" id="Lines_and_branches">Lines and
branches</span>, <span class="mw-headline" id="Rolling_stock">Rolling stock</span>, <span class="mw-
headline" id="Fares_and_ticketing">Fares and ticketing</span>, <span class="mw-headline"
id="Current_construction">Current construction</span>, <span class="mw-headline"
id="Opening_2020">Opening 2020</span>, <span class="mw-headline" id="Opening_2021">Opening 2021</span>,
<span class="mw-headline" id="Opening_2022">Opening 2022</span>, <span class="mw-headline"
id="Opening_2023">Opening 2023</span>, <span class="mw-headline" id="Tentative">Tentative</span>,
<span class="mw-headline" id="Under_planning">Under planning</span>, <span class="mw-headline"
id="Seoul_City">Seoul City</span>, <span class="mw-headline" id="Incheon_City">Incheon City</span>,
<span class="mw-headline" id="See_also">See also</span>, <span class="mw-headline"
id="Notes">Notes</span>, <span class="mw-headline" id="References">References</span>, <span class="mw-
headline" id="External_links">External links</span>]


[<span class="mw-headline" id="Overview">Overview</span>, <span class="mw-headline"
id="History">History</span>, <span class="mw-headline" id="Lines_and_branches">Lines and
branches</span>, <span class="mw-headline" id="Rolling_stock">Rolling stock</span>, <span class="mw-
headline" id="Fares_and_ticketing">Fares and ticketing</span>, <span class="mw-headline"
id="Current_construction">Current construction</span>, <span class="mw-headline"
id="Opening_2020">Opening 2020</span>, <span class="mw-headline" id="Opening_2021">Opening 2021</span>,
<span class="mw-headline" id="Opening_2022">Opening 2022</span>, <span class="mw-headline"
id="Opening_2023">Opening 2023</span>, <span class="mw-headline" id="Tentative">Tentative</span>,
<span class="mw-headline" id="Under_planning">Under planning</span>, <span class="mw-headline"
id="Seoul_City">Seoul City</span>, <span class="mw-headline" id="Incheon_City">Incheon City</span>,
<span class="mw-headline" id="See_also">See also</span>, <span class="mw-headline"
id="Notes">Notes</span>, <span class="mw-headline" id="References">References</span>, <span class="mw-
headline" id="External_links">External links</span>]
```

구글 뉴스 클리핑하기 1


```
import requests
from bs4 import BeautifulSoup

base_url = "https://news.google.com"
search_url = base_url +
"/search?q=%ED%8C%8C%EC%9D%B4%EC%8D%AC&hl=ko&gl=KR&ceid=KR%3Ako"
resp = requests.get(search_url)
html_src = resp.text
soup = BeautifulSoup(html_src, 'html.parser')

news_items = soup.select('div[class="xrnccd"]')
print(len(news_items))
print(news_items[0])
print("\n")

for item in news_items[:3]:
    link = item.find('a', attrs={'class':'VDXfz'}).get('href')
    news_link = base_url + link[1:]
    print(news_link)

    news_title = item.find('a', attrs={'class':'DY5T1d'}).getText()
    print(news_title)

    news_content = item.find('span', attrs={'class':'xBbh9'}).text
    print(news_content)
```

```
    news_agency = item.find('a', attrs={'class':'wEwyrc AVN2gc uQIVzc Sksgp'}).text
    print(news_agency)

    news_reporting = item.find('time', attrs={'class':'WW6dff uQIVzc Sksgp'})
    news_reporting_datetime = news_reporting.get('datetime').split('T')
    news_reporting_date = news_reporting_datetime[0]
    news_reporting_time = news_reporting_datetime[1][:-1]
    print(news_reporting_date, news_reporting_time)
    print("\n")

def google_news_clipping(url, limit=5):
    resp = requests.get(url)
    html_src = resp.text
    soup = BeautifulSoup(html_src, 'html.parser')

    news_items = soup.select('div[class="xrnccd"]')

    links=[]; titles=[]; contents=[]; agencies=[]; reporting_dates=[]; reporting_times=[];

    for item in news_items[:limit]:
        link = item.find('a', attrs={'class':'VDXfz'}).get('href')
        news_link = base_url + link[1:]
        links.append(news_link)
```

```
        news_title = item.find('a', attrs={'class':'DY5T1d'}).getText()
        titles.append(news_title)

        news_content = item.find('span', attrs={'class':'xBbh9'}).text
        contents.append(news_content)

        news_agency = item.find('a', attrs={'class':'wEwyrc AVN2gc uQIVzc Sksgp'}).text
        agencies.append(news_agency)

        news_reporting = item.find('time', attrs={'class':'WW6dff uQIVzc Sksgp'})
        news_reporting_datetime = news_reporting.get('datetime').split('T')
        news_reporting_date = news_reporting_datetime[0]
        news_reporting_time = news_reporting_datetime[1][:-1]
        reporting_dates.append(news_reporting_date)
        reporting_times.append(news_reporting_time)

    result = {'link':links, 'title':titles, 'contents':contents, 'agency':agencies, \
              'date':reporting_dates, 'time':reporting_times}

    return result

news = google_news_clipping(search_url, 2)
print(news)
```

구글 뉴스 클리핑하기 1

https://news.google.com/articles/CBMiI2h0dHA6Ly93d3cuY2lva29yZWEuY29tL25ld3MvMTU0MDEz0gEA?hl=ko&gl=KR&ceid=KR%3Ako
'줄리아 vs. 파이썬' 데이터 과학과 케미 좋은 언어는?
파이썬의 여러 응용 분야 가운데 아마도 데이터 애널리틱스가 가장 크고 중요할 것이다. 파이썬 진영에는 과학 컴퓨팅과 데이터 분석 작업을 신속하고 편리하게 ...
CIO Korea
2020-05-29 07:10:05

https://news.google.com/articles/CBMiJWh0dHA6Ly93d3cuaXR3b3JsZC5jby5rci9ob3d0by8xNTMxNDnSAQA?hl=ko&gl=KR&ceid=KR%3Ako
'속도를 높이는' 병렬 처리를 위한 6가지 파이썬 라이브러리
파이썬(Python)은 편의성과 프로그래머 친화성으로 유명하지만 속도 측면에서는 크게 내세울 것이 없는 프로그래밍 언어다. 파이썬의 속도 제약은 기본 구현인 c ...
ITWorld Korea
2020-05-20 07:45:17

https://news.google.com/articles/CBMiL2h0dHBzOi8vd3d3LnpkbmV0LmNvLmtyL3ZpZXcvP25vPTIwMjAwNTA1MTMzNDAy0gEA?hl=ko&gl=KR&ceid=KR%3Ako
마이크로소프트, 파이썬 무료 교육 영상 제공
마이크로소프트에서 제공하는 교육 영상 '초보자를 위한 파이썬'(이미지=마이크로소프트 디벨롭퍼 유튜브). 더보기+. ▷. "비만균 90% 녹이는물질 발견돼", 안빠지는 ...
ZD넷 코리아
2020-05-05 07:00:00

구글 뉴스 클리핑하기 2


```
import requests
from bs4 import BeautifulSoup
import urllib

keyword_input = '파이썬'
keyword = urllib.parse.quote(keyword_input)
print('파이썬 문자열을 URL 코드로 변환: ', keyword)

base_url = "https://news.google.com"
search_url = base_url + "/search?q=" + keyword + "&hl=ko&gl=KR&ceid=KR%3Ako"
print('검색어와 조합한 URL: ', search_url)

def google_news_clipping_keyword(keyword_input, limit=5):

    keyword = urllib.parse.quote(keyword_input)

    url = base_url + "/search?q=" + keyword + "&hl=ko&gl=KR&ceid=KR%3Ako"

    resp = requests.get(url)
    html_src = resp.text
    soup = BeautifulSoup(html_src, 'html.parser')

    news_items = soup.select('div[class="xrnccd"]')
    links=[]; titles=[]; contents=[]; agencies=[]; reporting_dates=[]; reporting_times=[];
```

구글 뉴스 클리핑하기 2

```
    for item in news_items[:limit]:
        link = item.find('a', attrs={'class':'VDXfz'}).get('href')
        news_link = base_url + link[1:]
        links.append(news_link)

        news_title = item.find('a', attrs={'class':'DY5T1d'}).getText()
        titles.append(news_title)

        news_content = item.find('span', attrs={'class':'xBbh9'}).text
        contents.append(news_content)

        news_agency = item.find('a', attrs={'class':'wEwyrc AVN2gc uQIVzc Sksgp'}).text
        agencies.append(news_agency)

        news_reporting = item.find('time', attrs={'class':'WW6dff uQIVzc Sksgp'})
        news_reporting_datetime = news_reporting.get('datetime').split('T')
        news_reporting_date = news_reporting_datetime[0]
        news_reporting_time = news_reporting_datetime[1][:-1]
        reporting_dates.append(news_reporting_date)
        reporting_times.append(news_reporting_time)

    result = {'link':links, 'title':titles, 'contents':contents, 'agency':agencies, \
              'date':reporting_dates, 'time':reporting_times}

    return result
```

구글 뉴스 클리핑하기 2


```
search_word = input("검색어를 입력하세요: ")
news = google_news_clipping_keyword(search_word, 2)
print(news['link'])
print(news['agency'])
```

```
파이썬 문자열을 URL 코드로 변환:  %ED%8C%8C%EC%9D%B4%EC%8D%AC
검색어와 조합한 URL:
https://news.google.com/search?q=%ED%8C%8C%EC%9D%B4%EC%8D%AC&hl=ko&gl=KR&ceid=KR%3Ako
검색어를 입력하세요: 파이썬
['https://news.google.com/articles/CBMiI2h0dHA6Ly93d3cuY2lva29yZWEuY29tL25ld3MvMTU0MDEz0gEA
?hl=ko&gl=KR&ceid=KR%3Ako',
'https://news.google.com/articles/CBMiJWh0dHA6Ly93d3cuaXR3b3JsZC5jby5rci9ob3d0by8xNTMxNDnSA
QA?hl=ko&gl=KR&ceid=KR%3Ako']
['CIO Korea', 'ITWorld Korea']
```

7. 웹 스크래핑

7.1 웹 스크래핑 I

7.2 웹 스크래핑 II

7.3 웹 스크래핑 III

```
from selenium import webdriver

driver = webdriver.Chrome("chromedriver")
driver.implicitly_wait(3)
driver.get("https://www.danawa.com/")

login = driver.find_element_by_css_selector('li.my_page_service > a')
print("HTML 요소: ", login)
print("태그 이름: ", login.tag_name)
print("문자열: ", login.text)
print("href 속성: ", login.get_attribute('href'))

login.click()
driver.implicitly_wait(3)

my_id = "----본인 아이디 입력하세요----"
my_pw = "----본인 패스워드 입력하세요----"

driver.find_element_by_id('danawa-member-login-input-id').send_keys(my_id)
driver.implicitly_wait(2)
driver.find_element_by_name('password').send_keys(my_pw)
driver.implicitly_wait(2)
driver.find_element_by_css_selector('button.btn_login').click()
```

```
HTML 요소:  <selenium.webdriver.remote.webelement.WebElement
(session="4e2721f3fb51d51d46c012660a07cc26", element="266bd1cc-aee9-4844-9068-
3f5d903767fd")>
태그 이름:  a
문자열:  로그인
href 속성:  https://auth.danawa.com/login?url=http%3A%2F%2Fwww.danawa.com%2F
```

동적 웹 페이지 2

```
from selenium import webdriver

driver = webdriver.Chrome("chromedriver")
driver.implicitly_wait(3)
driver.get("https://www.danawa.com/")

login = driver.find_element_by_css_selector('li.my_page_service > a')
login.click()
driver.implicitly_wait(3)

my_id = "----본인 아이디 입력하세요----"
my_pw = "----본인 패스워드 입력하세요----"

driver.find_element_by_id('danawa-member-login-input-id').send_keys(my_id)
driver.implicitly_wait(2)
driver.find_element_by_name('password').send_keys(my_pw)
driver.implicitly_wait(2)
driver.find_element_by_css_selector('button.btn_login').click()
driver.implicitly_wait(2)

wishlist = driver.find_element_by_css_selector('li.interest_goods_service > a').click()
driver.implicitly_wait(2)
html_src = driver.page_source
driver.close()
print(html_src[:])
```

동적 웹 페이지 2

```
from bs4 import BeautifulSoup
import re
soup = BeautifulSoup(html_src, 'html.parser')

wish_table = soup.select('table[class="tbl wish_tbl"]')[0]
wish_items = wish_table.select('tbody tr')

for item in wish_items:
    title = item.find('div', {'class':'tit'}).text
    price = item.find('span', {'class':'price'}).text
    link = item.find('a', href=re.compile('prod.danawa.com/info/')).get('href')
    print(title)
    print(price)
    print(link)
    print("\n")
```

```
Western Digital WD BLUE SN550 M.2 NVMe (500GB)
87,600원
http://prod.danawa.com/info/?pcode=10120452


삼성전자 860 EVO (500GB)
101,100원
http://prod.danawa.com/info/?pcode=5834210
```

한국은행 경제통계시스템 통계지표 활용하기 1


```
from selenium import webdriver
import time

def download_bok_statistics():

    driver = webdriver.Chrome("chromedriver")
    driver.implicitly_wait(3)
    driver.get("http://ecos.bok.or.kr/jsp/vis/keystat/#/key")


    excel_download = driver.find_element_by_css_selector('img[alt="download"]')
    driver.implicitly_wait(3)

    excel_download.click()
    time.sleep(5)

    driver.close()
    print("파일 다운로드 실행...")

    return None

download_bok_statistics()
```

```
파일 다운로드 실행...
```

한국은행 경제통계시스템 통계지표 활용하기 2

```
from selenium import webdriver
from bs4 import BeautifulSoup
import time

def download_bok_statistics_by_keyword():

    item_found = 0
    while not item_found:

        keyword = ""
        while len(keyword) == 0:
            keyword = str(input("검색할 항목을 입력하세요: "))

        driver = webdriver.Chrome("chromedriver")
        driver.implicitly_wait(3)
        driver.get("http://ecos.bok.or.kr/jsp/vis/keystat/#/key")
        time.sleep(5)

        items1 = driver.find_elements_by_css_selector('a[class="ng-binding"]')
        items2 = driver.find_elements_by_css_selector('a[class="a-c1-list ng-binding"]')
        items3 = driver.find_elements_by_css_selector('a[class="a-c4-list ng-binding"]')
        driver.implicitly_wait(3)

        items = items1[1:] + items2 + items3
```

```
        for idx, item in enumerate(items):
            if keyword in item.text:
                print("검색어 '%s'에 매칭되는 '%s' 통계지표를 검색 중..." % (keyword, item.text))
                item.click()
                item_found = 1
                time.sleep(5)
                break
            elif idx == (len(items) - 1):
                print("검색어 '%s'에 대한 통계지표가 존재하지 않습니다..." % keyword)
                driver.close()
                continue
            else:
                pass

    html_src = driver.page_source
    soup = BeautifulSoup(html_src, 'html.parser')
    driver.close()

    table_items = soup.find_all('td', {'class':'ng-binding'})
    date = [t.text for i, t in enumerate(table_items) if i % 3 == 0]
    value = [t.text for i, t in enumerate(table_items) if i % 3 == 1]
    change = [t.text for i, t in enumerate(table_items) if i % 3 == 2]

    result_file = open('bok_statistics_%s.csv' % keyword, 'w')
```

```
    for i in range(len(date)):
        result_file.write("%s, %s, %s" % (date[i], value[i], change[i]))
        result_file.write('\n')

    result_file.close()
    print("키워드 '%s'에 대한 통계지표를 저장하였습니다." % keyword)

    return date, value, change

result = download_bok_statistics_by_keyword()
print(result)
```

```
검색할 항목을 입력하세요: CD
검색어 'CD'에 매칭되는 'CD수익률(91일) ('20.05.29)' 통계지표를 검색 중...
키워드 'CD'에 대한 통계지표를 저장하였습니다.
(['2013', '2014', '2015', '2016', '2017', '2018', '2019', '2019.10', '2019.11', '2019.12',
'2020.1', '2020.2', '2020.3', '2020.4', '2020.5.22', '2020.5.25', '2020.5.26', '2020.5.27',
'2020.5.28', '2020.5.29'], ['2.72', '2.49', '1.76', '1.49', '1.44', '1.68', '1.69', '1.46',
'1.52', '1.53', '1.47', '1.42', '1.23', '1.10', '1.02', '1.02', '1.02', '1.02', '0.81',
'0.81'], ['-0.58', '-0.23', '-0.73', '-0.27', '-0.05', '0.24', '0.01', '-0.08', '0.06',
'0.01', '-0.06', '-0.05', '-0.19', '-0.13', '0.00', '0.00', '0.00', '0.00', '-0.21',
'0.00'])
```